FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algoritmos no dia-a-dia

Você certamente já utilizou um algoritmo, mesmo sem saber:

DESEMBALAR

1 Remover o material da embalagem, unidade por unidade.
  Nota: Guarde toda a embalagem.
2 Remova a sacola de plástico que cobre o rádio.
3 Remova o plástico que protege a tomada A/C.
4 Remover o nó da corda A/C e desatar fio da antena na parte traseira do rádio
5 Abrir a tampa e remover o material da embalagem de isopor da plataforma giratória.
6 Remover o lacre preto sob o braço.
7 Remova a tampa de proteção da agulha branca, puxando levemente para a frente da unidade.
8 Desatar a antena FM e deixe-o pendurado em uma linha reta para a recepção FM ideal.
  Se você tem problemas de sintonia em uma estação de FM, mova a antena externa para melhor recepção FM. Não ligar a antena FM para outra antena externa.

Mantenha-se à esquerda para continuar em Rod. Adalberto Panzan, siga as indicações para SP-348/Bandeirantes

7,1 km

Pegue a Rod. dos Bandeirantes
⚠ Estrada com pedágio em alguns trechos

68,1 km

Continue para Rod. dos Bandeirantes
⚠ Estrada com pedágio em alguns trechos

5,1 km

Continue em frente para permanecer na Rod. dos Bandeirantes

7,9 km

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Por exemplo, suponha que você está dirigindo em um bairro desconhecido e peça ajuda a alguém para chegar em determinado lugar.

Essa pessoa, provavelmente, irá lhe fornecer uma sequência de passos, do tipo "siga em frente por três ruas, depois vire à direita, siga por mais quatro ruas e vire à esquerda".

Talvez, em determinado ponto, você tenha que verificar se uma rua está ou não liberada para acesso e, caso estiver, seguir por ela, de forma a ser necessário analisar situações do ambiente variáveis para tomar decisão – em suma, realizar um raciocínio.

"Um algoritmo é um conjunto de regras, instruções e raciocínios que permitem chegar a um objetivo."

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Agora, imagine uma sequência de instruções presentes em um artigo científico a respeito de como isolar uma cultura de bactérias.

Evidentemente, seriam empregados diversos termos técnicos desconhecidos pelo público leigo, o que não seria um problema, uma vez que os leitores de tal artigo, provavelmente, já conhecem tais termos.

"As instruções de um algoritmo se adequam à pessoa ou público responsável por executá-lo"

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Por fim, voltemos ao exemplo que iniciou esta sessão. Dificilmente tais instruções viriam com detalhes do tipo "se o semáforo estiver vermelho, aguarde".

Isso acontece por que se espera que o leitor do algoritmo saiba como um semáforo funciona e como reagir diante dele, sendo desnecessário apresentar tais instruções.

Entretanto, um instrutor de direção provavelmente incluiria tais instruções.

"O nível de detalhamento do algoritmo também varia conforme a pessoa ou público responsável por executá-lo."

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Além disso, caso a pessoa que lhe apresentasse termos dúbios, tais como "vire para cá", você certamente iria desejar uma instrução mais clara.

Isso acontece por que um "algoritmo deve possuir passos claros o suficiente para não produzirem dúvida."

Por fim, é esperando que tal sequência de instruções termine – afinal, ninguém deseja ficar eternamente repetindo a mesma sequência de passos em círculos.

"É necessário que o algoritmo seja finito."

"Um algoritmo é um conjunto finito de instruções precisas que, se executadas, permitem a manipulação de um conjunto finito de dados de entrada para produzir um conjunto finito de dados de saída, dentro de um tempo finito."

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algoritmo na computação

Suponha um algoritmo para multiplicar dois números, revelar o resultado da operação e dizer se ele é negativo ou positivo. Até agora, temos algoritmos como meras sequências de instruções e, portanto, uma opção válida para construir tal algoritmo é a seguinte:

```
Obter os dois valores.

Multiplicar os dois valores

Se for positivo, informar que é positivo e se for negativo, informar que é negativo.

Informar o resultado
```

Porém, outra pessoa pode escrever o mesmo algoritmo usando termos diferentes. Ou, ainda, uma organização visual diferente.

Para um conjunto de passos pequeno, isso pode parecer funcional. Porém, para um algoritmo maior, com condições e repetições, tal técnica seria um verdadeiro caos.

Além disso, um algoritmo é criado visando sua posterior implementação em uma dada linguagem de programação. Como garantir que dois programadores diferentes, seguindo um mesmo algoritmo, não implementem códigos totalmente diferentes em uma mesma linguagem?

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Assim, na computação, o algoritmo deve ter:

1. Certo formalismo, normalmente convencionado previamente.

2. Certo padrão visual, voltado para facilitar a compreensão.

3. Um conjunto de estruturas lógicas que possuem equivalentes em linguagens convencionais, utilizadas para a criação dos procedimentos algorítmicos.

Algorithim Existência de raízes

Input: coeficientes e termo independente de uma equação de segundo grau

Output: se a equação possui ou não raízes reais

► Declaração das variáveis utilizadas

declare:

a,b,c: **integer**

discriminante: **real**

► Leitura dos dados de entrada

read a, b, c

► Cálculo do discriminante

discriminante ← $b^2 - 4 \times a \times c$

► Escrita dos dados de saída

if discriminante > 0 **then**

write "A equação de segundo grau possui raízes"

else

write "A equação de segundo grau não possui raízes"

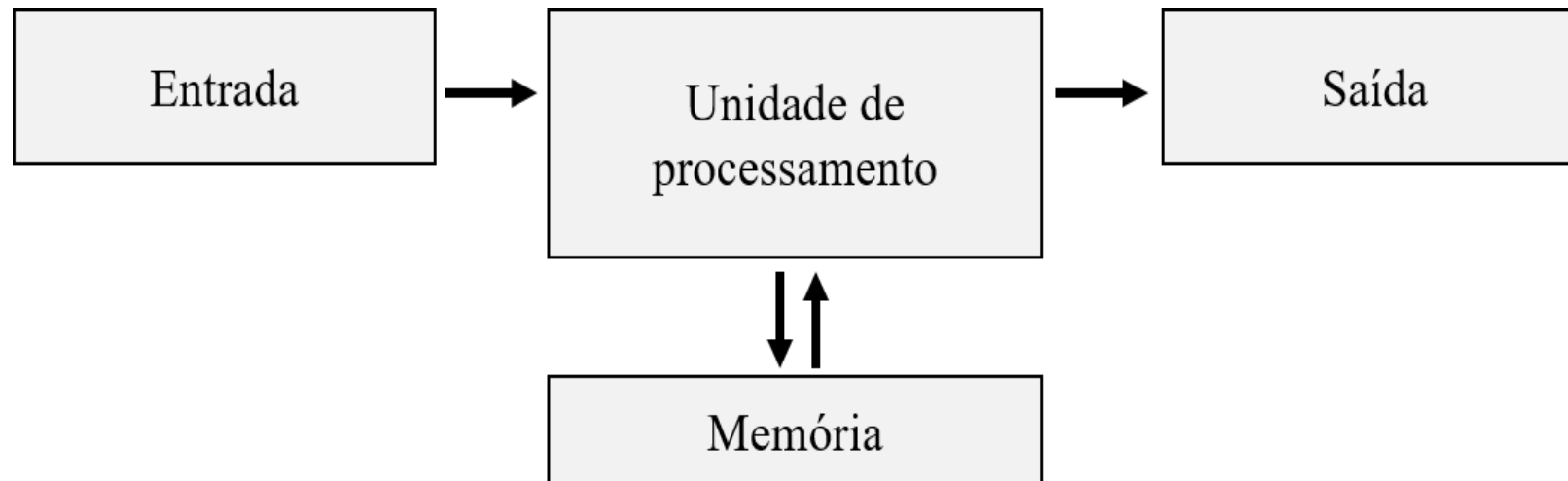
end if

end Algorithim

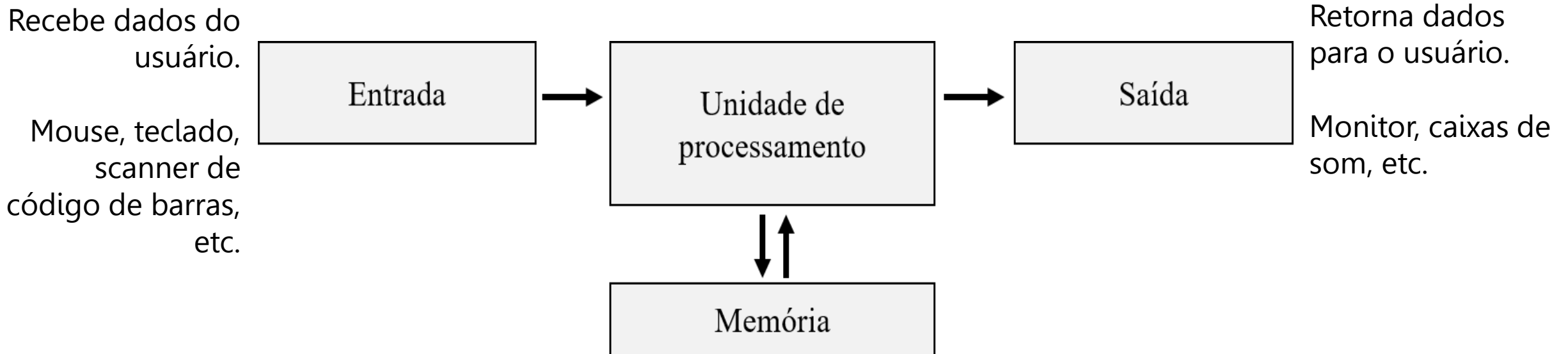
FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

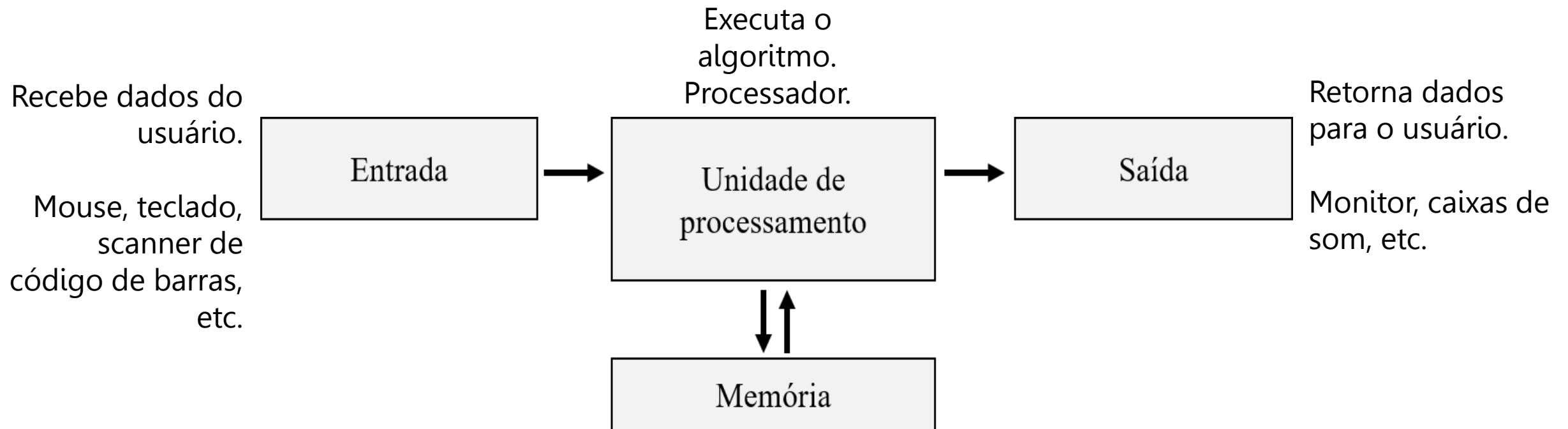
Modelo de computador

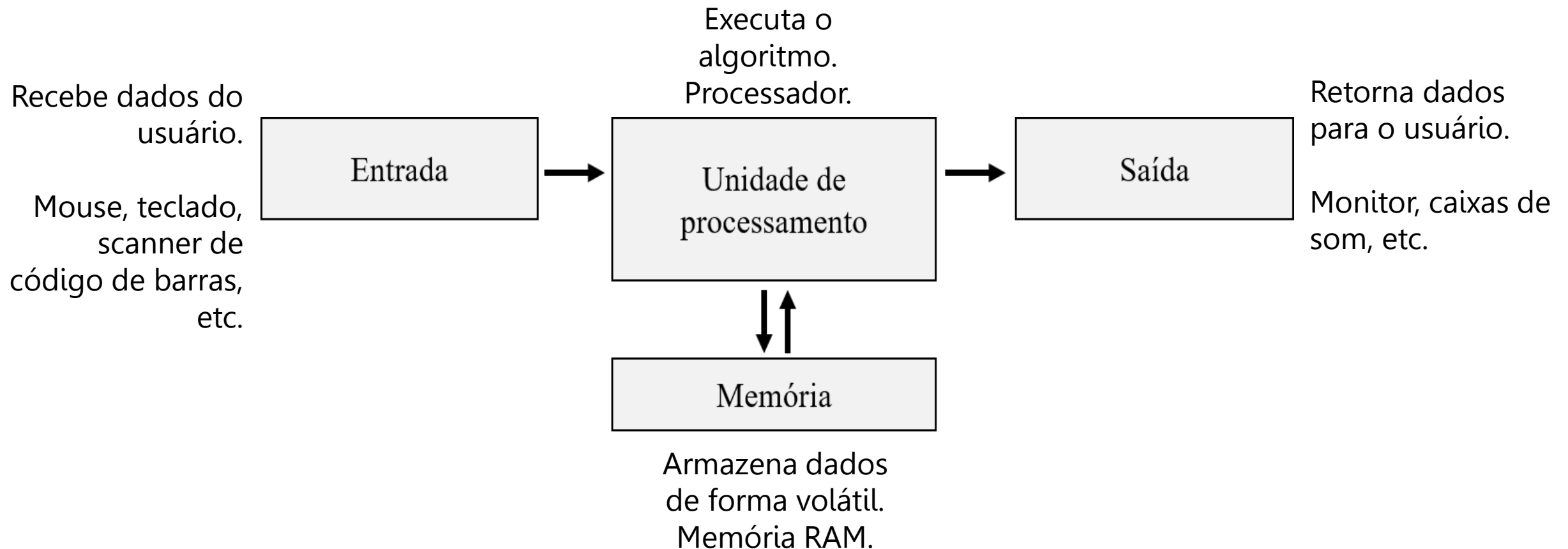
Os algoritmos criados nesse curso não serão executados em nenhum tipo de sistema real, mas sim mentalmente. Porém, eles são feitos visando sua execução em um computador real.

Assim, ao criar algoritmos, devemos considerar que eles serão executados em um modelo de computador, emulado mentalmente. Ele é composto por:

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Modelo de computador

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Modelo de computador

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Modelo de computador

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Variáveis

Suponha que, em um algoritmo, seja necessário receber dois valores e multiplica-los entre si, Além disso, é necessário elevar o resultado dessa multiplicação ao primeiro dos valores.

Receber os dois valores.

Multiplicá-los.

Elevar o resultado da multiplicação ao primeiro valor.

Informar o resultado da multiplicação e da potenciação na tela.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Variáveis

Suponha que, em um algoritmo, seja necessário receber dois valores e multiplica-los entre si, Além disso, é necessário elevar o resultado dessa multiplicação ao primeiro dos valores.

Evidentemente, chamar o primeiro e o segundo valor de valor_1 e valor_2, respectivamente, seria de grande ajuda para facilitar a compreensão do algoritmo.

Talvez, até pudéssemos chamar o resultado da multiplicação de multi, e o resultado da potência de exp.

Nesse exemplo, valor_1, valor_2, multi e exp foram nomes atribuídos a uma "estrutura" capaz de armazenar valores.

```
Receber valor_1 e valor_2

Fazer multi = valor_1 × valor_2

Fazer exp = multi ^ valor_1

Informar exp e multi
```

Variáveis

Suponha que, em um algoritmo, seja necessário receber dois valores e multiplica-los entre si, Além disso, é necessário elevar o resultado dessa multiplicação ao primeiro dos valores.

Evidentemente, chamar o primeiro e o segundo valor de valor_1 e valor_2, respectivamente, seria de grande ajuda para facilitar a compreensão do algoritmo.

Talvez, até pudéssemos chamar o resultado da multiplicação de multi, e o resultado da potência de exp.

Nesse exemplo, valor_1, valor_2, multi e exp foram nomes atribuídos a uma "estrutura" capaz de armazenar valores.

Variáveis

Receber valor_1 e valor_2

Fazer multi = valor_1 × valor_2

Fazer exp = multi ^ valor_1

Informar exp e multi

Variáveis

Receber valor_1 e valor_2

Fazer multi = valor_1 × valor_2

Fazer exp = multi ^ valor_1

Informar exp e multi

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Variáveis

Receber valor_1 e valor_2

Fazer multi = valor_1 × valor_2

Fazer exp = multi ^ valor_1

Informar exp e multi

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Variáveis

Receber valor_1 e valor_2

Fazer multi = valor_1 × valor_2

Fazer exp = multi ^ valor_1

Informar exp e multi

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Variáveis

Receber valor_1 e valor_2

Fazer multi = valor_1 × valor_2

Fazer exp = multi ^ valor_1

Informar exp e multi

2

valor_1

3

valor_2

6

multi

^

6

multi

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Variáveis

Receber valor_1 e valor_2

Fazer multi = valor_1 × valor_2

Fazer exp = multi ^ valor_1

Informar exp e multi

2
valor_1

3
valor_2

6
multi

64
exp

2
valor_1

^

6
multi

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Variáveis

Receber valor_1 e valor_2

Fazer multi = valor_1 × valor_2

Fazer exp = multi ^ valor_1

Informar exp e multi

Valor	Variável
2	valor_1
3	valor_2
6	multi
64	exp

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Variáveis

Receber valor_1 e valor_2

Fazer multi = valor_1 × valor_2

Fazer exp = multi ^ valor_1

Informar exp e multi

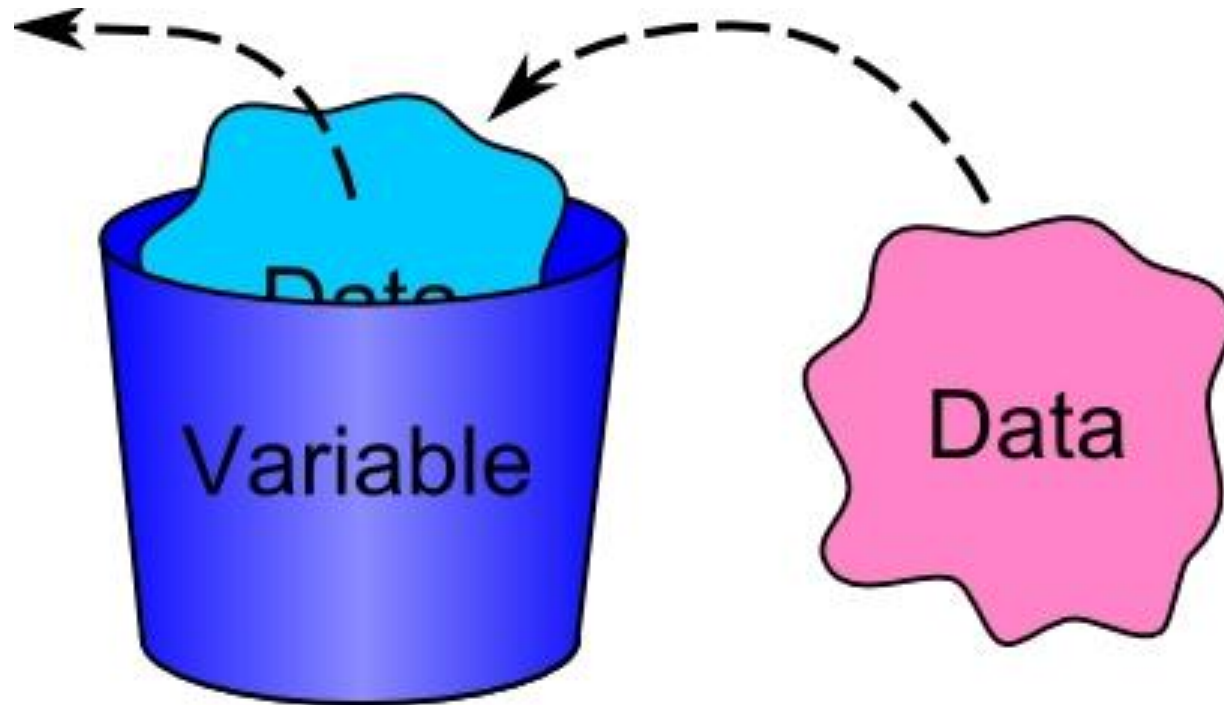
Valor	Variável
2	valor_1
3	valor_2
6	multi
64	exp

6

64

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Variáveis

"Na programação, uma variável é um objeto (uma posição, frequentemente localizada na memória) capaz de reter e representar um valor ou expressão. "

Tipos de dados

Até agora, falamos sobre a “caixa” (as variáveis). Mas o que podemos dizer sobre seu conteúdo, os dados?

Será que dados como:

1
8565
23,152
“Miojo”
“A”
true
false

Estão sujeitos às mesmas propriedades?

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Tipos de dados

Não estão! Eles se dividem em:

integer: como o próprio nome indica, uma variável do tipo inteiro pode receber números inteiros e permite a realização das operações no campo dos inteiros.

real: recebem números reais e permitem a realização de operações no campo dos reais.

string: para o armazenamento e manipulação de valores textuais, como nomes, códigos alfanuméricos e textos com e sem pontuação, Dessa forma, uma variável literal sempre corresponde a uma cadeia de caracteres.

boolean: outro tipo de dados bastante utilizado é o tipo lógico, que recebe apenas dois valores: verdadeiro ou falso, muitas vezes identificados como 1 e 0, respectivamente. Esse tipo de dado é bastante utilizado em condições, que serão estudadas posteriormente.

ponteiro: será estudado no futuro.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Tipos de dados

Até agora, falamos sobre a “caixa” (as variáveis). Mas o que podemos dizer sobre seu conteúdo, os dados?

Será que dados como:

1 (integer)
8565 (integer)
23,152 (real)
“Miojo” (string)
“A” (string)
true (boolean)
false (boolean)

Estão sujeitos às mesmas propriedades?

Refinamentos sucessivos

Quando nos deparamos com um problema muito extenso, é comum que não saibamos exatamente como começar a solução algorítmica para resolvê-lo. Nesse caso, muitos programadores recorrem à técnica de refinamentos sucessivos, isto é, o processo de escrever a mesma sequência de passos diversas vezes, mas cada vez mais refinada;

Refinar: tornar o código mais adequado ao padrão para escrita de algoritmos convencionado.

Exemplo: construa um algoritmo que leia os coeficientes a, b, e c da equação $ax^2 + bx + c$ e informe quantas raízes reais ela possui.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Ler os valores dos coeficientes e do termo independente

Calcular o discriminante

Com base no sinal do discriminante, informa quantas raízes existem.

Ler a, b, c sendo a e b os coeficientes e c o termo independente

$determinante = b^2 - 4 * a * c$

Se determinante ≥ 0:

    Dizer que tem raízes reais

Se determinante < 0:

    Dizer que não tem raízes reais

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algorithim Existência de raízes

Input: coeficientes e termo independente de uma equação de segundo grau

Output: se a equação possui ou não raízes reais

► Declaração das variáveis utilizadas

    declare:

        a,b,c: **integer**

        discriminante: **real**

► Leitura dos dados de entrada

    read a, b, c

► Cálculo do discriminante

    discriminante ← b^2 - 4 × a × c

► Escrita dos dados de saída

    if discriminante > 0 **then**

        write "A equação de segundo grau possui raízes"

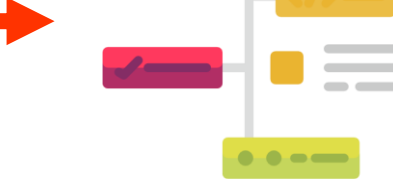
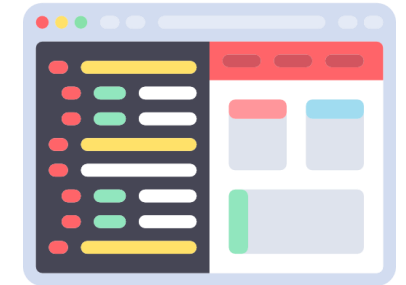
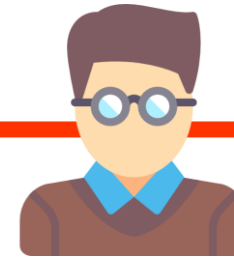
    else

        write "A equação de segundo grau não possui raízes"

    end if

end Algorithim

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Qual a função de um algoritmo?

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Cabeçalho

Algorithim: Fatorial de um número

Input: número inteiro

Output: fatorial desse número

Algorithim: Lista de candidatos

Input: nome e nota de *n* candidatos

Output: quais candidatos foram aprovados

Rodapé

end Algorithim

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

```
Algorithim Existência de raízes
Input: coeficientes e termo independente de uma equação de segundo grau
Output: se a equação possui ou não raízes reais
▶ Declaração das variáveis utilizadas
        declare:
                a,b,c: integer
                discriminante: real
▶ Leitura dos dados de entrada
        read a, b, c
▶ Cálculo do discriminante
        discriminante ← b² - 4 × a × c
▶ Escrita dos dados de saída
        if discriminante > 0 then
                write "A equação de segundo grau possui raízes"
        else
                write "A equação de segundo grau não possui raízes"
        end if
end Algorithim
```

Cabeçalho

Corpo

Rodapé

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Comentários

```
▶ Escrita dos dados de saída

    if discriminante > 0 then
        write "A equação de segundo grau possui raízes"
    else
```

```
▶ Contagem até 10

    for i ← 0 to 10 step 1 do
        a ← a + 1 ▶ Adiciona 1 ao valor de a
        write a
    end for
```

Declaração de variáveis

```
declare:
    a,b,c, discriminante: integer
```

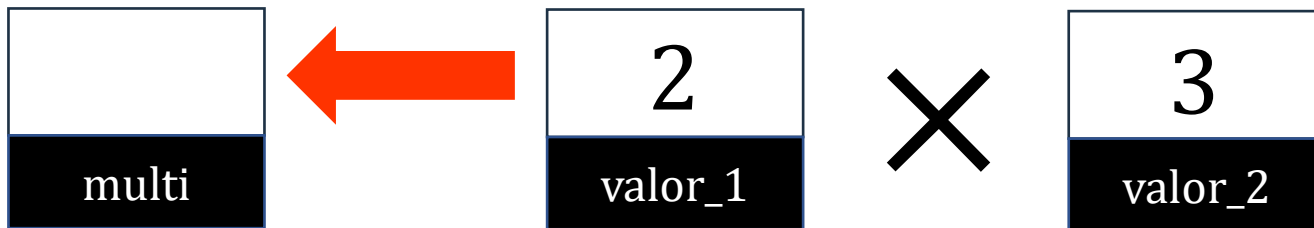
```
declare:
    a: integer
    b: real
```

```
declare:
    nome: string
    nota: real
    aprovado: boolean
    total_alunos: integer
    i: integer ► contador
```

Para uma variável ser utilizada no código, ela antes deve ser declarada.

Declarar uma variável equivale a informar qual seu identificador (ou seja, o nome da variável) e qual seu tipo, conforme os tipos de dados estudados na aula anterior.

Atribuição de variáveis

A atribuição é o processo de "dar um valor" para uma variável. O operador de atribuição é ← e ele é utilizado sempre que uma variável recebe um valor oriundo da execução do código (e não inserido pelo usuário).

```
a ← -2

b ← 5.3
```

```
soma_valores ← Ø
```

```
nome ← "Fábrica de Noobs"
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Atribuição de variáveis

A atribuição é o processo de “dar um valor” para uma variável. O operador de atribuição é ← e ele é utilizado sempre que uma variável recebe um valor oriundo da execução do código (e não inserido pelo usuário).

multi ← valor_1 × valor_2

Atribuição de variáveis

A atribuição é o processo de "dar um valor" para uma variável. O operador de atribuição é ← e ele é utilizado sempre que uma variável recebe um valor oriundo da execução do código (e não inserido pelo usuário).

```
multi ← valor_1 × valor_2
```

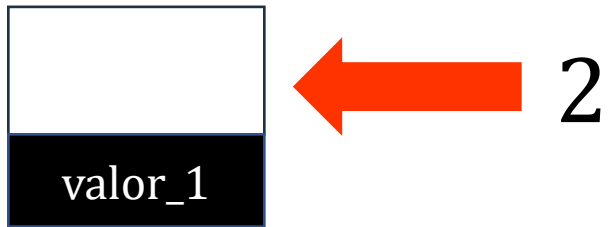
Leitura de variáveis

A leitura é o processo de “dar um valor inserido pelo usuário” para uma variável.

```
read a
```

```
read nota_1, nota_2, nota_3
```

Leitura de variáveis

A leitura é o processo de "dar um valor inserido pelo usuário" para uma variável.

read valor_1

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Escrita

A escrita é o processo de exibir um valor para o usuário.

write *"Morte ao Miojoo"*

write media, frequência

- Aqui, a parte "e, portanto, ele está aprovado" não está indicada explicitamente e, portanto, deve ser indicada.

write *"a média do aluno é"*, media, *"com frequência de"*, frequência, *"e, portando, ele está aprovado"*

- Aqui, o texto não é necessário, uma vez que não contém nenhuma informação adicional.

write *"a média do aluno é"*, media, *"com frequência de"*, frequência

- Portanto, deve-se usar a sintaxe:

write media, frequencia

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Operadores matemáticos

Função	Recebe	Retorna	Descrição
a+b	Real	Real	Adição
a-b	Real	Real	Subtração
a/b	Real	Real	Divisão
a×b	Real	Real	Multiplicação
a ^ b	Real	Real	Potenciação
a++	Real	Real	Incremento
a--	Real	Real	Decremento

A = A - 1

A--

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Funções matemáticas

Função	Recebe	Retorna	Descrição
sen (r), cos (r), tan(r), asen(r), acos(r), atan(r)	Real	Real	Funções trigonométricas usuais
ln (r), log (r)	Real	Real	Funções logarítmicas usuais
trunca(r)	Real	Inteiro	Retorna a parte inteira de um número real
arred(r)	Real	Inteiro	Retorna o inteiro mais próximo do real passado
abs(r)	Real	Real	Módulo (valor absoluto) de um número real
resto(r,i)	Real/inteiro	Inteiro	Retorna o resto da divisão entre dois números
quoc(r,i)	Real/inteiro	Inteiro	Retorna o quociente da divisão entre dois números
raiz(r)	Real/inteiro	Real	Raiz quadrada de um número

Funções de texto

Função	Recebe	Retorna	Descrição
s + r	String	String	Concatena duas strings.
comprLiteral(s)	String	Inteiro	Comprimento de uma string.
valLiteral(s)	String	Real	Retorna o valor numérico de uma cadeia de caracteres.

```
A <- “25”
valLiteral(a)
```

Outras funções

Preciso usar uma função que não está listada! O que fazer?

1. Usar a função normalmente no texto, mas inserir um comentário explicando qual seu papel. Faça isso sempre que o detalhamento da função fugir do escopo do algoritmo.

1. Criar uma sub-rotina, com o conteúdo visto nas aulas futuras;

Outras funções

Preciso usar uma função que não está listada! O que fazer?

1. Usar a função normalmente no texto, mas inserir um comentário explicando qual seu papel. Faça isso sempre que o detalhamento da função fugir do escopo do algoritmo.

1. Criar uma sub-rotina, com o conteúdo visto nas aulas futuras;

Proposições

Definimos uma proposição como uma sentença declarativa, que pode assumir valores-verdade V (indicando que a sentença é verdadeira) ou F (indicando que a sentença é falsa).

O que isso quer dizer? Em termos leigos, toda afirmação que pode ser verdadeira ou falsa é uma preposição. Por exemplo, “2 é maior que 3”, “a palavra “fabrica” tem 7 letras”, “hoje faz sol” são preposições, pois podem ser verdadeiras ou falsas.

Na maioria dos casos, só faz sentido utilizar uma proposição como parâmetro para executar ou não uma ação se seu valor verdade for variável (ou seja, pode ser V ou F). Logo, nada mais sensato do que estruturarmos nossas proposições com base em uma relação entre uma ou mais variáveis, a qual pode ser, portanto, verdadeira ou falsa.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

São proposições

5 + 3 = 8

“O Palmeiras não tem mundial”

8 > 3 ou 10 < 100

$200 - 30 \neq 170$

“Todos os números são primos ou compostos”

Não são proposições

8

“Bom dia”

“Palmeiras”

Operadores relacionais

Operador	Função
$=$	Igual
$\neq$	Diferente
$>$	Estritamente maior
$<$	Estritamente menor
$\geq$	Maior ou igual
$\leq$	Menor ou igual

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algorithim Valores verdade

Input: não há.

Output: não há.

declare:

a, b: **integer**

r,s: **boolean**

$2 + 2 = 4$ ► V

$2 + 2 > 4$ ► F

$2 + 2 \geq 4$ ► V

$a \leftarrow 2$

$b \leftarrow 5$

$2 + 2 = 4$ ► V

$a + b = 7$ ► V

"batata" = "batata" ► V

compLiteral("batata") = compLiteral("teste") ► F

$r \leftarrow$ **true**

$s \leftarrow$ **false**

$s = r$ ► F

$s \neq r$ ► V

Operadores relacionais

Operador	Função
=	Igual
≠	Diferente
>	Estritamente maior
<	Estritamente menor
≥	Maior ou igual
≤	Menor ou igual

Operadores lógicos

Operador	Função
not	Não
and	E
or	Ou

Operadores lógicos

Proposição	Retorno
not(V)	F
not(F)	V

Operadores lógicos

Proposição	Retorno
V and V	V
V and F	F
F and V	F
F and F	F

M >= 6 and F >= 75%

M = 0,5 and F = 30%

F and F

F

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Operadores lógicos

Proposição	Retorno
V or V	V
V or F	V
F or V	V
F or F	F

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algorithim Valores verdade

Input: não há.

Output: não há.

declare:

a, b: **integer**

r,s: **boolean**

$2 + 2 = 4$ **and** $2 + 2 = 5$ ► F

$2 + 2 > 4$ **or** $2 + 2 = 4$ ► V

$a \leftarrow 2$

$b \leftarrow 5$

$a = b$ **or** $a \neq b$ ► V

$a = b$ **and** $a \neq b$ ► F

not $(a = b)$ **or** **not**$(a \neq b)$ ► V

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Construa um programa que receba dois valores e diga se o produto entre eles é positivo ou negativo"

Algorithim Sinal do produto de dois valores

Input: dois valores inteiros.

Output: se o produto entre eles é positivo ou negativo

declare:

a, b, produto: **integer**

read a,b

produto ← a × b

Como indicar o programa que ele deve verificar se o produto é ou não negativo?

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Vamos voltar à técnica de refinamentos sucessivos e escrever o programa em uma linguagem menos formal.

Receber os números a e b.

Realizar $produto = a \times b$

Se $produto \geq 0$:

    Informar que o resultado é positivo

Se não:

    Informar que o resultado é negativo

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Você concorda que $produto \geq 0$ é uma proposição?
E que a estrutura que criamos realiza uma operação conforme o valor–verdade dessa proposição?

Receber os números a e b.

Realizar $produto = a \times b$

proposição

Se $produto \geq 0$: → Se proposição tem valor-verdade V

Informar que o resultado é positivo

Se não: → Se a proposição tem valor-verdade F

Informar que o resultado é negativo

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Assim, iremos introduzir uma nova estrutura: a estrutura if.

Algorithm Sinal do produto de dois valores

Input: dois valores inteiros.

Output: se o produto entre eles é positivo ou negativo

declare:

  a, b, produto: **integer**

read a,b

produto ← a × b

if produto ≥ 0 **then**

  write "*O produto entre os números é positivo*"

else

  write "*O produto entre os números é negativo*"

end if

end Algorithm

if produto ≥ 0 **then**

  write "*O produto entre os números é positivo*"

else

  write "*O produto entre os números é negativo*"

end if

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Assim, iremos introduzir uma nova estrutura: a estrutura if.

Algorithm Sinal do produto de dois valores

Input: dois valores inteiros.

Output: se o produto entre eles é positivo ou negativo

declare:

a, b, produto: **integer**

read a,b

produto ← a × b

proposição

if produto ≥ 0 **then** → Se proposição tem valor-verdade V

write "*O produto entre os números é positivo*"

else → Se a proposição tem valor-verdade F

write "*O produto entre os números é negativo*"

end if

end Algorithm

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Uma variável recebe um número digitado pelo usuário, que pode ser 1,2, ou 3. Se o resultado for 1, o programa exibe a mensagem "Palmeiras", se for 2, exibe "Internacional" e, se for 3, exibe "Flamengo". Se não for nenhum desses valores, exibe "Não encontrado"."

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algorithm Três primeiros colocados no campeonato brasileiro no dia 08/10/2018

Input: número de 1 a 3.

Output: time correspondente à colocação inserida.

declare:

  menu: **integer**

read menu

if menu = 1 **then**

  write *"Palmeiras"*

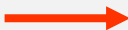
else if menu = 2 **then**

  write *"Internacional"*

else if menu = 3 then

  write *"Flamengo"*

else

  write *"Não encontrado"*

end if

end Algorithm

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algorithm Três primeiros colocados no campeonato brasileiro no dia 08/10/2018

Input: número de 1 a 3.

Output: time correspondente à colocação inserida.

declare:

menu: **integer**

read menu

if menu = 1 **then** → Se a proposição tiver valor-verdade V

write *"Palmeiras"*

else if menu = 2 **then** → Se a proposição tiver valor-verdade V

write *"Internacional"*

else if menu = 3 then → Se a proposição tiver valor-verdade V

write *"Flamengo"*

else → Se nenhuma das proposições anteriores tiverem valor-verdade V

write *"Não encontrado"*

end if

end Algorithm

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

```
declare:
        menu: integer
read menu
switch menu of
        case 1 do
                write "Palmeiras"
        end case
        case 2 do
                write "Internacional"
        end case
        case 3 do
                write "Flamengo"
        end case
        otherwise do
                write "Não encontrado"
        end otherwise
end switch
end Algorithm
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

```
declare:
    menu: integer
read menu
switch menu of
    case 1 do        → Se a proposição menu = 1 tiver valor-verdade V
        write "Palmeiras"
    end case
    case 2 do        → Se a proposição menu = 2 tiver valor-verdade V
        write "Internacional"
    end case
    case 3 do        → Se a proposição menu = 3 tiver valor-verdade V
        write "Flamengo"
    end case
    otherwise do     → Se nenhuma das proposições anteriores tiverem valor-verdade V
        write "Não encontrado"
    end otherwise
end switch
end Algorithm
```

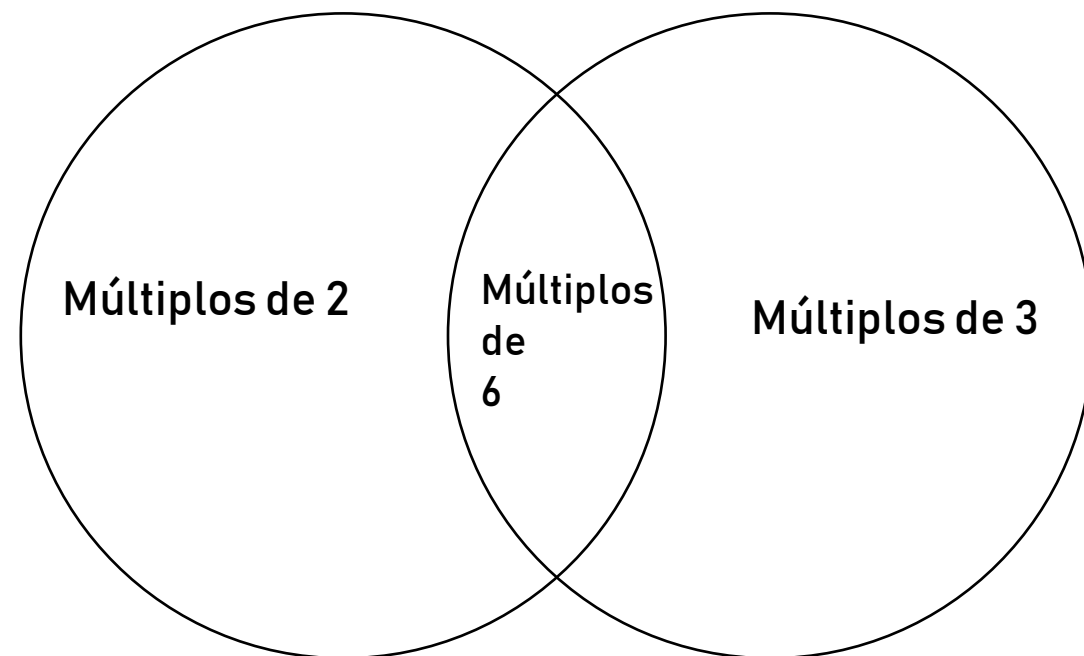
FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Faça um programa que informe se um número é múltiplo de 2, de 3 ou de ambos."

Nesse caso, usar qualquer uma das estruturas acima sem se importar com a ordem em que elas aparecem poderia causar problemas.

Por exemplo, se optássemos por verificar se um número é múltiplo de 2 antes de verificar se ele é múltiplo de 6, correríamos o risco de não dizer que 12 é múltiplo de 6.

Para situações do tipo, não há um processo único e totalmente funcional, mas sim recomendações que você pode seguir.

A principal delas é usar comandos if aninhados, colocando as possibilidades menos gerais dentro de outras, mais gerais.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

```
if  resto (numero,2) = 0 then

        write "Multiplo de 2"

else if resto(numero, 3) = 0 then

        write "Multiplo de 3 "

else if resto (numero, 6) = 0 then

        write "Multiplo de 6"

end if
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algorithm Múltiplo de 2,3 ou 6

Input: número inteiro.

Output: se o número inserido é múltiplo de 2,3 ou 6

declare:

numero: **integer**

read numero

if resto(numero, 6) = 0 **then**

write *"O número é múltiplo de 6"*

write "*O número é múltiplo de 3"*

write "O número é múltiplo de 2"

else

if resto(numero, 3) = 0 **then**

write *"O número é múltiplo de 3"*

else if resto(numero, 2) = 0 **then**

write *"O numero é múltiplo de 2"*

end if

end if

end Algorithm

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algorithm Múltiplo de 2,3 ou 6
Input: número inteiro.

Output: se o número inserido é múltiplo de 2,3 ou 6

declare:

numero: **integer**

read numero

if resto(numero, 6) = 0 **then**

write *"O número é múltiplo de 6"*

write *"O número é múltiplo de 3"*

write "O número é múltiplo de 2"

else

if resto(numero, 3) = 0 **then**

write *"O número é múltiplo de 3"*

else if resto(numero, 2) = 0 **then**

write *"O numero é múltiplo de 2"*

end if

end if

end Algorithm

Condição menos geral

Condições mais gerais, de mesma hierarquia

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Construa um programa que imprima os números de 1 até 10".

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Construa um programa que imprima os números de 1 até 10".

Definir um contador i

Assinalar $i = 1$

Se $i \leq 10$:

  Imprimir i
  Acrescentar i em uma unidade
  Repetir a terceira linha

Se não:

  Encerrar o algoritmo

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Construa um programa que imprima os números de 1 até 10".

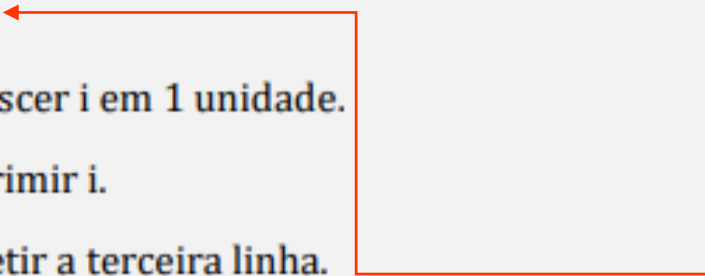
Definir um contador i.

Assinalar i = 0.

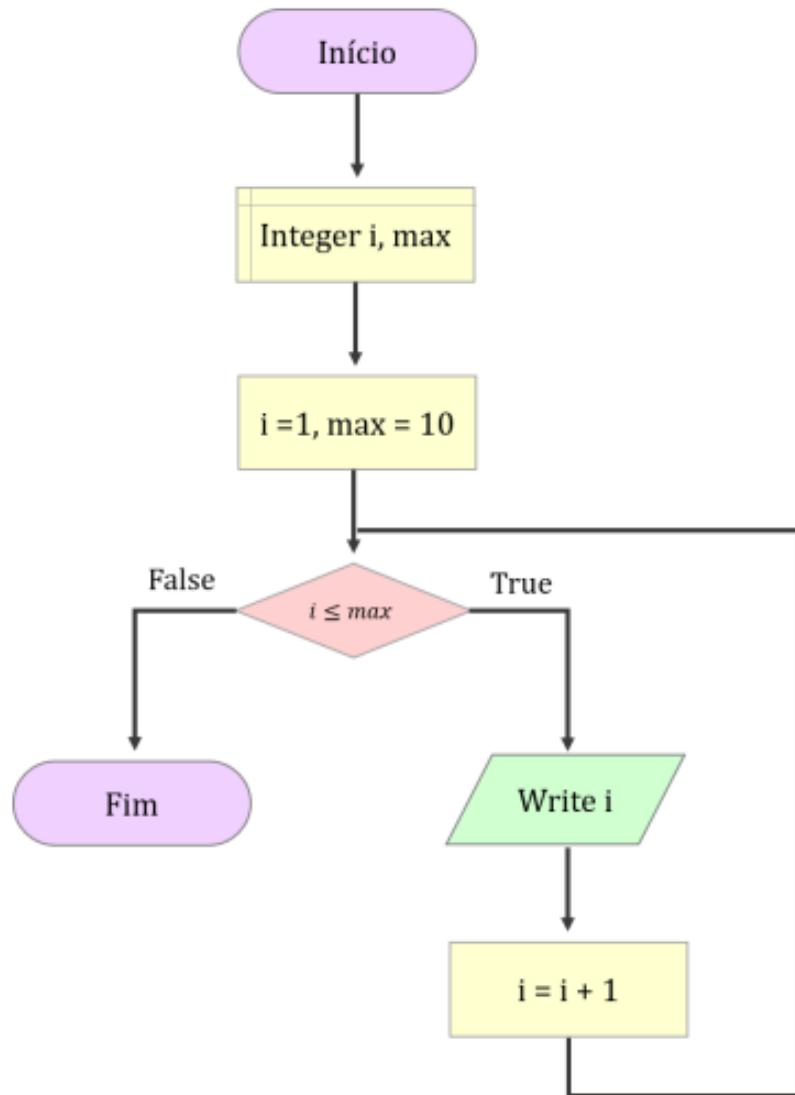
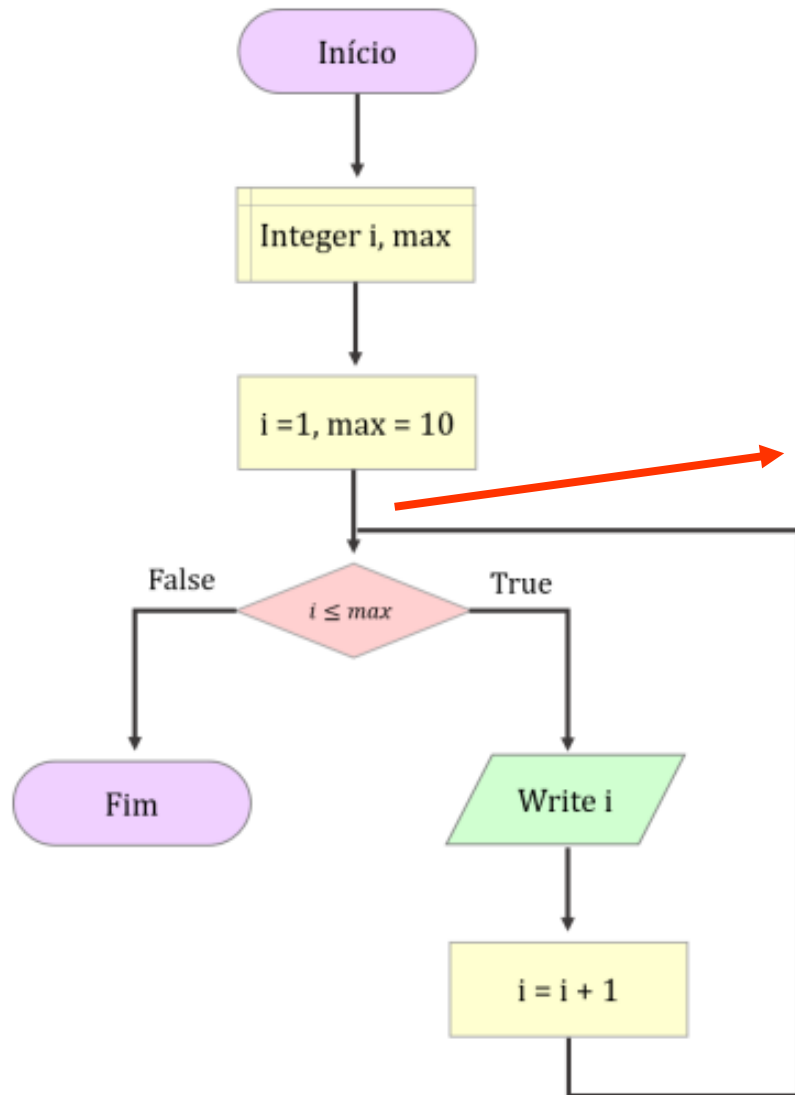
Se $i \leq 10$:

    Acrescer i em 1 unidade.

    Imprimir i.

    Repetir a terceira linha.

Se não:

    Encerrar o algoritmo.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Porém, não podemos fazer tal operação em diagrama de blocos ou em algoritmo.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Para contornar o problema, podemos adotar o laço for.

```
for i ← 1 to 10 step 1 do
        ▶ Comando a ser executado em cada vez
end for
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Para contornar o problema, podemos adotar o laço for.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algorithm: contar até dez

Input: não há.

Output: não há.

for i ← 1 **to** 10 **step** 1 **do**

        write i

end for

end Algorithm

O resultado da compilação será:

1
2
3
4
5
6
7
8
9
10

```
a = 10

for i <-  1 to (a + 200) step 1 do

        > Codigo
        a = 300

End for
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

1. Caso o valor inicial seja maior que o valor final, os comandos internos não serão executados nenhuma vez e o valor da variável de controle será simplesmente igual ao valor de início (não haverá incremento, pois não houve execução);

2. Se o valor de início for igual ao valor de fim, os comandos serão executados uma única vez e a variável de controle terminará com valor incrementado de 1;

3. Em pseudocódigo, não é necessário declarar previamente a variável de controle, por mais que o Flowgorithm e algumas linguagens de programação exijam isso;

4. Não se pode alterar o valor da variável de controle dentro do laço;

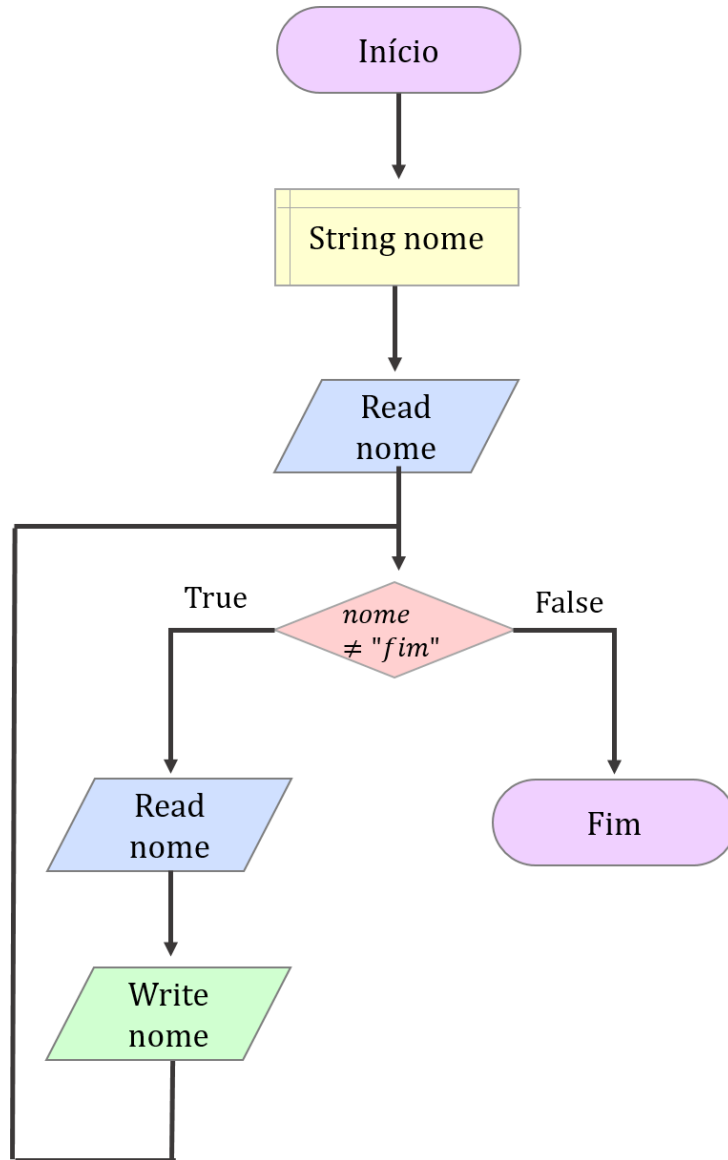
5. Caso os valores de início e de fim sejam formados por expressões que usam variáveis, estas expressões serão avaliadas somente uma vez antes das repetições começarem; portanto, se houver modificação dos valores das variáveis, o número de repetições estimado inicialmente não será alterado.

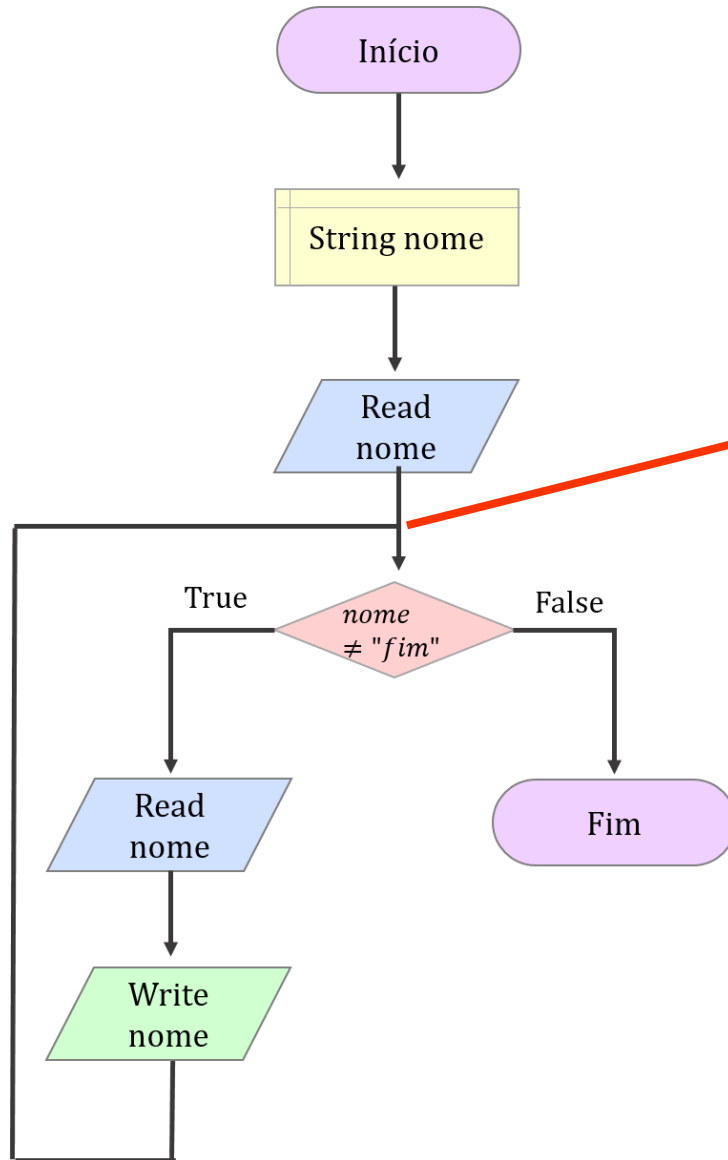
FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Crie um programa que recebe uma lista de nomes e os imprima na tela seguidos de uma saudação, até que o nome inserido seja "fim". Nesse caso, encerre o programa".

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Crie um programa que recebe uma lista de nomes e os imprima na tela seguidos de uma saudação, até que o nome inserido seja "fim". Nesse caso, encerre o programa".

```
Declarar a variável nome

Solicitar a inserção de um nome

Se nome ≠ "fim"

        Imprimir o nome
        Solicitar a inserção de um nome
        Voltar para a terceira linha

Se nome = "fim"

        Encerrar o algoritmo
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Porém, não podemos fazer tal operação em diagrama de blocos ou em algoritmo.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Proposição

while nome ≠ fim **do**

▶ Comando a ser executado enquanto a expressão nome ≠ fim tiver valor-verdade V

end while

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algorithim: nome de pessoas
Input: nome de pessoas
Output: saudação para essas pessoas

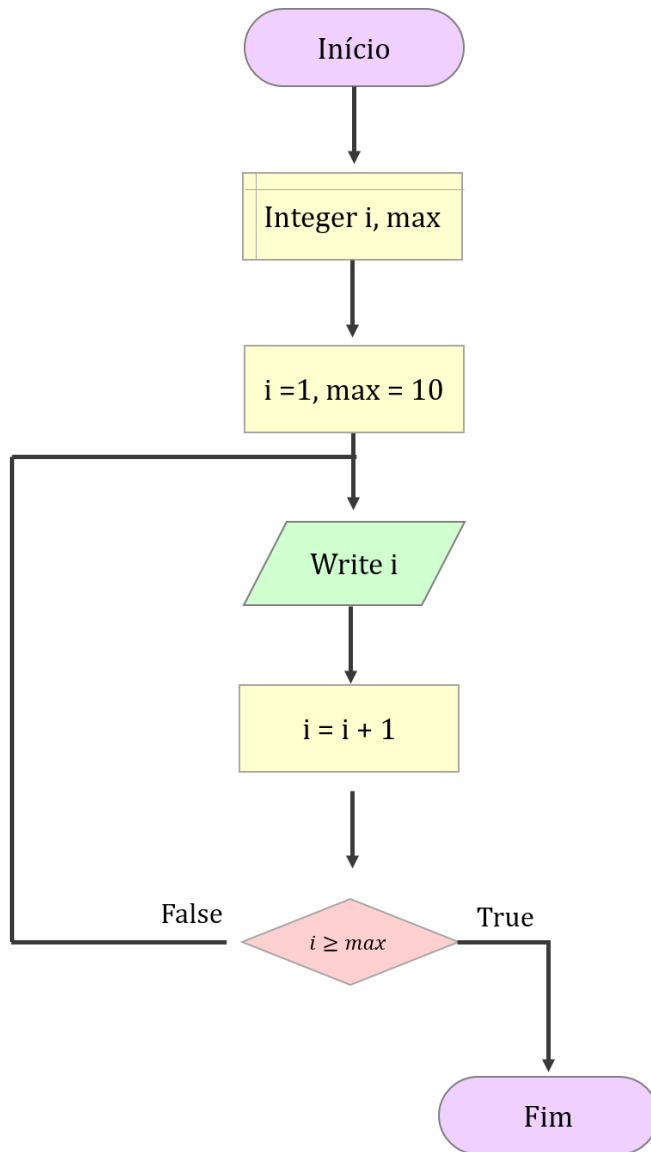
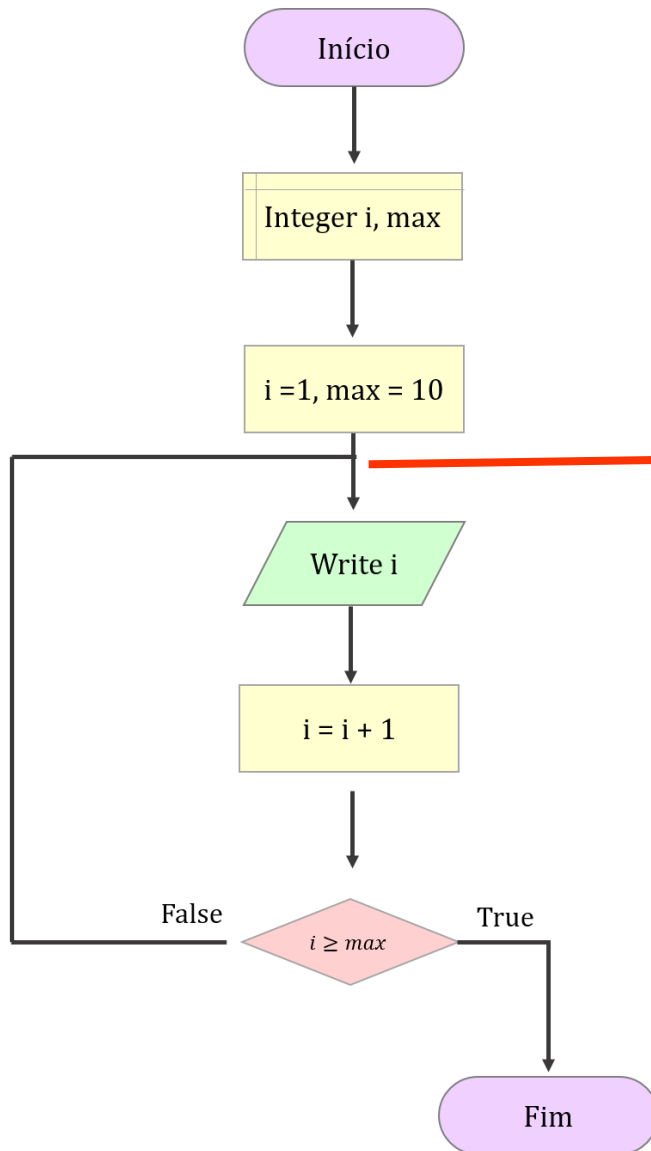
declare:
    nome: **string**

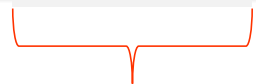
read nome

while nome ≠ fim **do**

    write "Ola", nome
    read nome

end while

end algorithim

Início
Integer i, max
i =1, max = 10
Write i
i = i + 1
False
$i \geq max$
True
Fim

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

repeat
▶ Comando a ser executado enquanto a expressão nome = fim tiver valor-verdade
F
until $i \geq 10$
Proposição

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algorithim: contar até dez
Input: não há
Output: números de 1 até 10

declare:
    i, max: **integer**

i ← 1
max ← 10

repeat
    write i
    $i = i + 1$

until i ≤ max

end algorithim

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Em primeiro lugar, é extremamente comum, para estudantes iniciados, escrever programas que produzem um loop infinito quando executados, o que o ocorre por que a condição que deve encerrar o loop nunca ocorre. Para evitar que isso aconteça, deve-se revisar o código e "testá-lo" mentalmente, visando encontrar situações nas quais ele não terá fim.

Em segundo lugar, deve-se lembrar que as expressões que condicionam o fim do laço em while e repeat são diferentes umas das outras (afinal de contas, para o primeiro caso, o laço continua se a expressão tiver valor-verdade V e, para o segundo caso, se a expressão tiver valor-verdade F).

Por fim, para comandos while, é possível que um conjunto de comandos não seja executado nenhuma vez (caso a condicional nunca se verifique). Já para comandos repeat, esse conjunto sempre será executado ao menos uma única vez, uma vez que a verificação da condição só ocorre depois da execução do conjunto de comandos.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Construa um programa que receba as três notas de um aluno em uma disciplina. Em seguida, o programa deve imprimir um boletim, contendo cada uma das notas e a média final."

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Construa um programa que receba as três notas de um aluno em uma disciplina. Em seguida, o programa deve imprimir um boletim, contendo cada uma das notas e a média final."

Algorithm: boletim

Input: notas de um aluno em 3 provas.

Output: boletim contendo nota de cada prova e sua média final

declare:

    nota1, nota2,nota3: **real**

    media: **real**

read nota1, nota2, nota3

media ← (nota1 + nota2 + nota3) / 3

write nota1, nota2, nota3

write media

end Algorithm

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

E se fôssemos capazes de armazenar todas as n notas em uma mesma variável?

```
Algorithm: boletim
Input: notas de um aluno em 3 provas.
Output: boletim contendo nota de cada prova e sua média final

declare:
        nota1, nota2,nota3: real
        media: real
read nota1, nota2, nota3
media ← (nota1 + nota2 + nota3) / 3
write nota1, nota2, nota3
write media
end Algorithm
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

E se fôssemos capazes de armazenar todas as n notas em uma mesma variável?

Algorithm: boletim

Input: notas de um aluno em 3 provas.

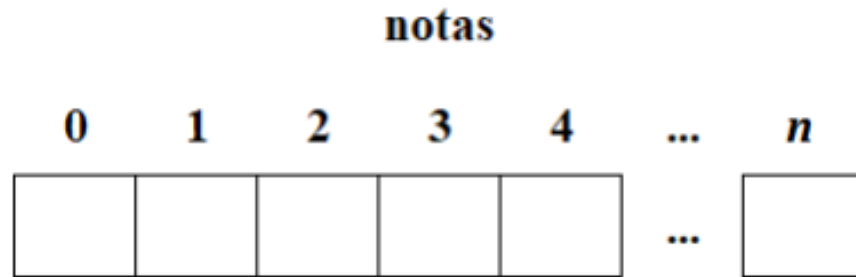
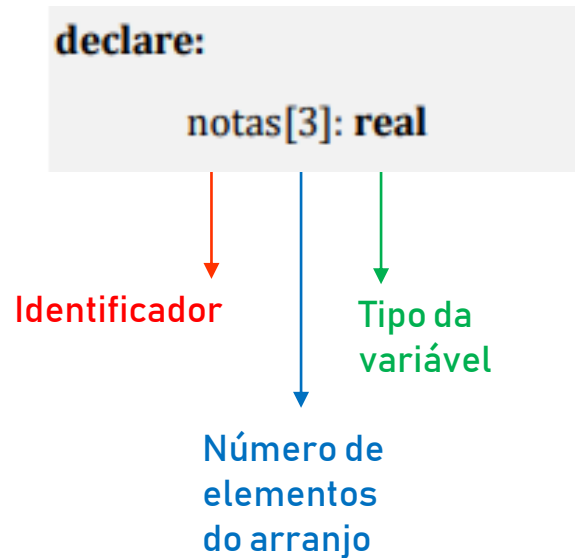
Output: boletim contendo nota de cada prova e sua média final

```
declare:
        nota1, nota2,nota3: real
        media: real
read nota1, nota2, nota3
media ← (nota1 + nota2 + nota3) / 3
write nota1, nota2, nota3
write media
end Algorithm
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

E se fôssemos capazes de armazenar todas as n notas em uma mesma variável?

E se fôssemos capazes de armazenar todas as n notas em uma mesma variável?

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

E se fôssemos capazes de armazenar todas as n notas em uma mesma variável?

read notas[2]

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Atribuição

```
temp ← notas[2]
```

```
notas[2]← notas[2] + 2
```

Percorrer todo o arranjo

```
for i → 0 to 2 step 1 do
        read notas[i]
        media ← media + notas[i]
end for
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Comando a ser executado em cada elemento do arranjo

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

```
Algorithm: boletim
Input: notas de um aluno em 3 provas.
Output: boletim contendo nota de cada prova e sua média final
► Declaração das variáveis utilizadas
declare:
        notas[3]: real
        media: real
► Leitura das notas
        media ← 0
► Leitura das notas
for i → 0 to 2 step 1 do
        read notas[i]
        media ← media + notas[i]
end for
► Determinação da média
media ← media / 3
► Escrita dos resultados
for i → 0 to 2 step 1 do
        write notas[i]
end for
write media
end Algorithm
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Construa um programa que receba os elementos de uma matriz com n linhas e m colunas, os quais podem ser, no máximo, 50. Em seguida, o programa deve exibir a matriz na tela."

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Construa um programa que receba os elementos de uma matriz com n linhas e m colunas, os quais podem ser, no máximo, 50. Em seguida, o programa deve exibir a matriz na tela."

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Construa um programa que receba os elementos de uma matriz com n linhas e m colunas, os quais podem ser, no máximo, 50. Em seguida, o programa deve exibir a matriz na tela."

declare:

matriz[20][30]:**real**

temp ← matriz[3][2] ▸ atribui o elemento presente na linha 3 e coluna 2 a uma variável

write matriz [0][1] ▸ escreve na tela o elemento presente na linha 0 e coluna 1

write matriz[5][5] ▸ escreve na tela o elemento presente na linha 5 e coluna 5

write matriz[a][b] ▸ escreve na tela o elemento presente na linha de valor igual a variável a, e na coluna de valor igual a variável b

write matriz[a + 1][b − 2] ▸ escreve na tela o elemento presente na linha de valor igual a variável a + 1, e na coluna de valor igual a variável b - 2

```
Algorithm: matriz
Input: número de linhas e colunas de uma matriz, elementos da matriz
Output: a própria matriz inserida
► Declaração das variáveis utilizadas
declare:
    matriz[50][50]: real
    num_linhas: integer
    num_colunas: integer
► Recebimento da quantidade de linhas e colunas
    read num_linhas
    read num_colunas
► Recebimento dos elementos da matriz
    for i → 0 to num_linhas - 1 step 1 do
        for j → 0 to num_colunas - 1 step 1 do
            read matriz[i][j]
        end for
    end for
► Impressão da matriz
    for i → 0 to num_linhas - 1 step 1 do
        for j → 0 to num_colunas - 1 step 1 do
            write matriz[i][j]
        end for
    end for
end Algorithm
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

declare:

estoque[2][4][4][3]: **integer**

- 1. Estoque de uma loja de calças. A primeira dimensão corresponde ao sexo (0 = masculino, 1 = feminino), a segunda dimensão corresponde ao tamanho (0,1,2,3), a terceira dimensão corresponde ao comprimento (0,1,2,3), e a quarta dimensão corresponde a cor (0 = azul, 1 = cinza, 2 = preto)

observacao[32][13][11]: **integer**

- 2.A primeira dimensão corresponde ao dia do ano, a segunda ao mês, e a terceira ao ano (que assume valores de 1 a 10). Aqui, optamos por desprezar os índices 0, e começar a contar a partir do 1.

pontoemR4[99][99][99][99]: **real**

- 3.Cada dimensão corresponde a uma coordenada de um vetor no espaço vetorial $\mathbb{R}^4$, que contém índices até 100 em cada dimensão. Se você não sabe álgebra linear, não se preocupe em entender o contexto desse exemplo.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

declare:

estoque[2][4][4][3]: **integer**

- 1. Estoque de uma loja de calças. A primeira dimensão corresponde ao sexo (0 = masculino, 1 = feminino), a segunda dimensão corresponde ao tamanho (0,1,2,3), a terceira dimensão corresponde ao comprimento (0,1,2,3), e a quarta dimensão corresponde a cor (0 = azul, 1 = cinza, 2 = preto)

observacao[32][13][11]: **integer**

- 2.A primeira dimensão corresponde ao dia do ano, a segunda ao mês, e a terceira ao ano (que assume valores de 1 a 10). Aqui, optamos por desprezar os índices 0, e começar a contar a partir do 1.

pontoemR4[99][99][99][99]: **real**

- 3.Cada dimensão corresponde a uma coordenada de um vetor no espaço vetorial $\mathbb{R}^4$, que contém índices até 100 em cada dimensão. Se você não sabe álgebra linear, não se preocupe em entender o contexto desse exemplo.

8
15/11/5

observacao [30][2][5]

declare:

estoque[2][4][4][3]: **integer**

- 1. Estoque de uma loja de calças. A primeira dimensão corresponde ao sexo (0 = masculino, 1 = feminino), a segunda dimensão corresponde ao tamanho (0,1,2,3), a terceira dimensão corresponde ao comprimento (0,1,2,3), e a quarta dimensão corresponde a cor (0 = azul, 1 = cinza, 2 = preto)

observacao[32][13][11]: **integer**

- 2.A primeira dimensão corresponde ao dia do ano, a segunda ao mês, e a terceira ao ano (que assume valores de 1 a 10). Aqui, optamos por desprezar os índices 0, e começar a contar a partir do 1.

pontoemR4[99][99][99][99]: **real**

- 3.Cada dimensão corresponde a uma coordenada de um vetor no espaço vetorial $\mathbb{R}^4$, que contém índices até 100 em cada dimensão. Se você não sabe álgebra linear, não se preocupe em entender o contexto desse exemplo.

declare:

estoque[1][4][4][2]

0 homem
1 mulher

0 P
1 M
2 G
3 GG

0 50 cm
1 60 cm
2 70 cm
3 80 cm

0 azul
1 cinza
2 preto

```
estoque[0][0][2][0] <- 5

write estoque[0][0][2][0]
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Um estudante deseja registrar a quantidade de peixes presentes em um aquário durante 10 anos. Crie um programa que receba essa quantidade para cada dia do mês (desprezando anos bissextos, mas considerando a quantidade de dias em cada mês) e depois, conforme inserido pelo usuário, informe o total de peixes em um dia, mês e ano específico."

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algorithm: número de peixes em um aquário

Input: quantidade de peixes em cada dia, durante um período de 5 anos, dia desejado para receber essa quantidade

Output: quantidade de peixes no dia desejado

▶ Declaração das variáveis utilizadas

declare:

total_peixes[32][13][6]: **integer**

input_dia: **integer**

input_mes: **integer**

input_ano: **integer**

quantidade_dias[13]: **integer**

▶ Criação do vetor que indica quanto dias há em cada mês

quantidade_dias[1] ← 31

quantidade_dias[2] ← 28

quantidade_dias[3] ← 31

quantidade_dias[4] ← 30

quantidade_dias[5] ← 31

quantidade_dias[6] ← 30

quantidade_dias[7] ← 31

quantidade_dias[8] ← 31

quantidade_dias[9] ← 30

quantidade_dias[10] ← 31

quantidade_dias[11] ← 30

quantidade_dias[12] ← 31

▶ Recebimento da quantidade de peixes em cada dia

for a → 1 **to** 10 **step** 1 **do**

for m → 1 **to** 12 **step** 1 **do**

for d → 1 **to** quantidade_dias[m] **step** 1 **do**

read total_peixes[d][m][a]

end for

end for

end for

▶ Escrita da quantidade de peixes em um dado dia

read input_dia, input_mes, input_ano

write total_peixes[input_dia][input_mes][input_ano]

end Algorithm

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Uma estação meteorológica coleta, durante um período de 50 dias, dados relativos a umidade relativa do ar (número real), a temperatura (número real) e o tipo de nuvem (texto). Construa um programa que recebe todas essas informações e apresente a temperatura média observada nesse período."

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Uma estação meteorológica coleta, durante um período de 50 dias, dados relativos a umidade relativa do ar (número real), a temperatura (número real) e o tipo de nuvem (texto). Construa um programa que recebe todas essas informações e apresente a temperatura média observada nesse período."

```
declare:
        umidade[50]: real
        temperatura[50]: real
        nuvem[50]: string
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Uma estação meteorológica coleta, durante um período de 50 dias, dados relativos a umidade relativa do ar (número real), a temperatura (número real) e o tipo de nuvem (texto). Construa um programa que recebe todas essas informações e apresente a temperatura média observada nesse período."

```
declare:
        umidade[50]: real
        temperatura[50]: real
        nuvem[50]: string
```

	0	1	2	3	4	...	49
umidade						...	
temperatura						...	
nuvem						...	

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Seria interessante se pudéssemos criar um novo tipo de variável, chamada, por exemplo, de dia que possui como campo todas as demais variáveis, umidade, temperatura, nuvem.

Seria interessante se pudéssemos criar um novo tipo de variável, chamada, por exemplo, de dia que possui como campo todas as demais variáveis, umidade, temperatura, nuvem. Assim, cada variável do tipo dia poderia ser ilustrada como se fosse uma ficha, como mostrado abaixo.

dia
umidade: **real**
temperatura: **real**
nuvem: **string**

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Seria interessante se pudéssemos criar um novo tipo de variável, chamada, por exemplo, de dia que possui como campo todas as demais variáveis, umidade, temperatura, nuvem. Assim, cada variável do tipo dia poderia ser ilustrada como se fosse uma ficha, como mostrado abaixo.

dia
umidade: real
temperatura: real
nuvem: string

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Seria interessante se pudéssemos criar um novo tipo de variável, chamada, por exemplo, de dia que possui como campo todas as demais variáveis, umidade, temperatura, nuvem. Assim, cada variável do tipo dia poderia ser ilustrada como se fosse uma ficha, como mostrado abaixo.

dia
umidade: **real**
temperatura: **real**
nuvem: **string**

```
type dia: struct
        umidade: real
        temperatura: real
        nuvem: string
    end struct
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

```
declare:
        natal: dia
        pascoa: dia
```

Cada variável do tipo dia

```
umidade: real
temperatura: real
nuvem: string
```

dia
umidade: **double**
temperatura: **double**
nuvem: **string**

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

```
declare:
        natal: dia
        pascoa: dia
```

Cada variável do tipo dia

```
umidade: real
temperatura: real
nuvem: string
```

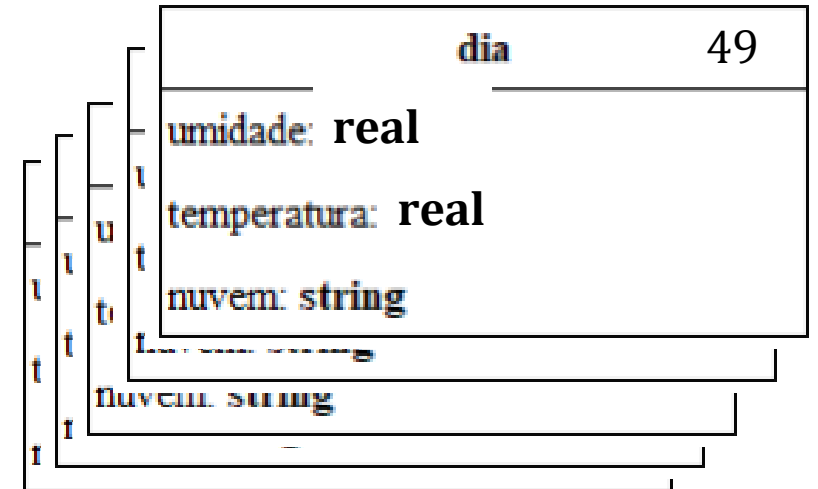
```
natal.temperatura ← 30
natal.nuvem← "cirrus"
natal.umidade ←17
pascoa.temperatura ←32
write natal.temperatura + pascoa.temperatura
```

dia
umidade: real temperatura: real nuvem: string

```
declare:
        natal: dia
        pascoa: dia
```

Cada variável do tipo dia

```
umidade: real
temperatura: real
nuvem: string
```

```
dias[50]: dia
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

```
Algorithm: observações meteorológicas durante 50 dias do ano
Input: umidade do ar, temperatura e tipo de nuvem de 50 dias do ano.
Output: temperatura média nesse período
▶Declaração da estrutura
    type dia: struct
            umidade: real
            temperatura: real
            nuvem: string
        end struct
▶Declaração das demais variáveis
    declare:
        dias[50]: dia
        media: real
▶Definição do valor padrão
    media ← 0
▶ Recebimento dos valores de entrada
    for i ← 0 to 49 step 1 do
        read dias[i].umidade
        read dias[i].temperatura
        read dias[i].nuvem
    end for
▶ Cálculo da média
    for i ← 0 to 49 step 1 do
        media ← media + dias[i].temperatura
    end for
    media ← media /50
    write media
end Algorithm
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Sabemos que as variáveis utilizadas pelo algoritmo são armazenadas em parte da memória, e que essas variáveis possuem um identificador – que permitem que elas sejam chamadas durante o algoritmo.

Porém, além do identificador, cada parte da memória também possui um endereço.

integer	Tipo do dado. Indica a forma como o dado será armazenado na memória.
ano_nascimento	Identificador da variável, utilizado para "chamá-la" durante o algoritmo.
1999	Valor assumido pela variável, e utilizado nas operações.
endereço: #0x61ff1c	Endereço da variável. Indica o local na memória em que a variável em questão está alocada.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Sabemos que as variáveis utilizadas pelo algoritmo são armazenadas em parte da memória, e que essas variáveis possuem um identificador – que permitem que elas sejam chamadas durante o algoritmo.

Porém, além do identificador, cada parte da memória também possui um endereço.

integer	Tipo do dado. Indica a forma como o dado será armazenado na memória.
ano_nascimento	Identificador da variável, utilizado para "chamá-la" durante o algoritmo.
1999	Valor assumido pela variável, e utilizado nas operações.
endereço: #0x61ff1c	Endereço da variável. Indica o local na memória em que a variável em questão está alocada.

Uma variável do tipo ponteiro armazena esse endereço, na forma de uma instrução para acessar tal endereço na memória.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

nome: string

PTR_nome: ^string

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

PTR_nome <- &nome
PTR_nome <- &ano_nascimento

endereço_ano_nascimento ← &ano_nascimento

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algorithm: exemplo do uso de ponteiros

Input: não há

Output: valor da variável, após ser alterada pelo ponteiro

▶ Declaração das variáveis

declare:

    valor: **integer**

    PTR_valor: ↑**integer**

▶ Atribuição do endereço da variável valor para o ponteiro

    PTR_valor ← &valor ▶ O ponteiro PTR_valor aponta para a variável valor

▶ Modificação da variável valor via ponteiro

    ↑PTR_valor ← 241

▶ Escrita dos dados de saída

    write valor

▶ O output é 241.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

▶ Neste exemplo, temos um arranjo dado por arranjo[4]:integer e queremos o endereço da variável arranjo[2]

▶ A primeira possibilidade é:

endereço_arranjo ← &arranjo[0] + 2

▶ A segunda possibilidade é:

endereço_arranjo ← &arranjo[2]

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Crie um programa que trabalha com um novo tipo de dado, denominado vetor, que é um vetor em $\mathbb{R}^2$, com coordenadas x e y. Em seguida, crie dois vetores, de coordenadas $A = (1,3)$ e $B = (0,2)$.

Some $A + B$ e, com isso, crie um novo vetor C.

Em seguida, some $C + A$, e coloque esse resultado em outro vetor, D.

Por fim, calcule a norma dos vetores D e A.

Em seguida, exiba na tela o valor dos vetores A, B, C e D, além das normas calculadas.

Além disso, considere que:

$$(a, b) + (c, d) = (a + c, b + d)$$

$$\|(a, b)\| = \sqrt{a^2 + b^2}$$

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algorithm: manipulações com vetores em $\mathbb{R}^2$

Input: vetores A e B

Output: soma dada por C = A + B, soma dada por D = C + A, norma do vetor D e do vetor A

```
► Criação do tipo de dado vetor
type vetor: struct
        x: integer
        y: integer
    end struct
► Declaração das variáveis utilizadas
declare:
    A: vetor
    B: vetor
    C: vetor
    D: vetor
    normaA: real
    normaD: real
► Recebimento dos valores de cada vetor
    A.x ← 1
    A.y ← 3
    B.x ← 0
    B.y ← 2
► Cálculo da soma dos vetores
    C.x ← A.x + B.x
    C.y ← A.y + B.y
    D.x ← A.x + C.x
    D.y ← A.y + Cy
► Cálculo da norma dos vetores
    normaA ← raiz((A.x)^2 +( A.y)^2)
    normaD ← raiz((D.x)^2 +( D.y)^2)
► Escrita dos dados de saída
    write "(", A.x, ",", "A.y, ")"
    write "(", B.x, ",", "B.y, ")"
    write "(", C.x, ",", "C.y, ")"
    write "(", D.x, ",", "D.y, ")"
    write normaA
    wrtie normaD
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

```
Algorithm: manipulações com vetores em ℝ²
Input: vetores A e B
Output: soma dada por C = A + B, soma dada por D = C + A, norma do vetor D e do vetor A
▶ Criação do tipo de dado vetor
type vetor: struct
        x: integer
        y: integer
    end struct
▶ Declaração das variáveis utilizadas
declare:
    A: vetor
    B: vetor
    C: vetor
    D: vetor
    normaA: real
    normaD: real
▶ Recebimento dos valores de cada vetor
    A.x ← 1
    A.y ← 3
    B.x ← 0
    B.y ← 2
```

Atribui valores para as coordenadas x e y.

```
▶ Cálculo da soma dos vetores
    C.x ← A.x + B.x
    C.y ← A.y + B.y
    D.x ← A.x + C.x
    D.y ← A.y + Cy
▶ Cálculo da norma dos vetores
    normaA ← raiz((A.x)^2 +( A.y)^2)
    normaD ← raiz((D.x)^2 +( D.y)^2)
▶ Escrita dos dados de saída
    write "(", A.x, ",", "A.y, ")"
    write "(", B.x, ",", "B.y, ")"
    write "(", C.x, ",", "C.y, ")"
    write "(", D.x, ",", "D.y, ")"
    write normaA
    wrtie normaD
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algorithm: manipulações com vetores em $\mathbb{R}^2$

Input: vetores A e B

Output: soma dada por C = A + B, soma dada por D = C + A, norma do vetor D e do vetor A

```
▶ Criação do tipo de dado vetor
type vetor: struct
        x: integer
        y: integer
    end struct
▶ Declaração das variáveis utilizadas
declare:
    A: vetor
    B: vetor
    C: vetor
    D: vetor
    normaA: real
    normaD: real
▶ Recebimento dos valores de cada vetor
    A.x ← 1
    A.y ← 3
    B.x ← 0
    B.y ← 2
```

Atribui valores para as coordenadas x e y.

```
▶ Cálculo da soma dos vetores
    C.x ← A.x + B.x
    C.y ← A.y + B.y
    D.x ← A.x + C.x
    D.y ← A.y + Cy
```

$(a, b) + (c, d) = (a + c, b + d)$

```
▶ Cálculo da norma dos vetores
    normaA ← raiz((A.x)^2 +( A.y)^2)
    normaD ← raiz((D.x)^2 +( D.y)^2)
▶ Escrita dos dados de saída
    write "(", A.x, ",", "A.y, ")"
    write "(", B.x, ",", "B.y, ")"
    write "(", C.x, ",", "C.y, ")"
    write "(", D.x, ",", "D.y, ")"
    write normaA
    wrtie normaD
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algorithm: manipulações com vetores em $\mathbb{R}^2$

Input: vetores A e B

Output: soma dada por C = A + B, soma dada por D = C + A, norma do vetor D e do vetor A

▶ Criação do tipo de dado vetor

type vetor: **struct**

x: **integer**

y: **integer**

end **struct**

▶ Declaração das variáveis utilizadas

declare:

A: **vetor**

B: **vetor**

C: **vetor**

D: **vetor**

normaA: **real**

normaD: **real**

▶ Recebimento dos valores de cada vetor

A.x ← 1

A.y ← 3

B.x ← 0

B.y ← 2

Atribui valores para as coordenadas x e y.

▶ Cálculo da soma dos vetores

C.x ← A.x + B.x

C.y ← A.y + B.y

D.x ← A.x + C.x

D.y ← A.y + Cy

$$(a, b) + (c, d) = (a + c, b + d)$$

▶ Cálculo da norma dos vetores

normaA ← **raiz**((A.x)^2 +(A.y)^2)

normaD ← **raiz**((D.x)^2 +(D.y)^2)

$$\|(a, b)\| = \sqrt{a^2 + b^2}$$

▶ Escrita dos dados de saída

write "(", A.x, ",", "A.y, ")"

write "(", B.x, ",", "B.y, ")"

write "(", C.x, ",", "C.y, ")"

write "(", D.x, ",", "D.y, ")"

write normaA

wrtie normaD

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Algorithm: manipulações com vetores em $\mathbb{R}^2$

Input: vetores A e B

Output: soma dada por C = A + B, soma dada por D = C + A, norma do vetor D e do vetor A

```
▶ Criação do tipo de dado vetor
type vetor: struct
            x: integer
            y: integer
        end struct
▶ Declaração das variáveis utilizadas
declare:
      A: vetor
      B: vetor
      C: vetor
      D: vetor
      normaA: real
      normaD: real
▶ Recebimento dos valores de cada vetor
      A.x ← 1
      A.y ← 3
      B.x ← 0
      B.y ← 2
```

Atribui valores para as coordenadas x e y.

```
▶ Cálculo da soma dos vetores
      C.x ← A.x + B.x
      C.y ← A.y + B.y
      D.x ← A.x + C.x
      D.y ← A.y + Cy
```

$(a, b) + (c, d) = (a + c, b + d)$

```
▶ Cálculo da norma dos vetores
      normaA ← raiz((A.x)^2 +( A.y)^2)
      normaD ← raiz((D.x)^2 +( D.y)^2)
```

$\|(a, b)\| = \sqrt{a^2 + b^2}$

```
▶ Escrita dos dados de saída
      write "(", A.x, ",", "A.y, ")"
      write "(", B.x, ",", "B.y, ")"
      write "(", C.x, ",", "C.y, ")"
      write "(", D.x, ",", "D.y, ")"
      write normaA
      wrtie normaD
```

Informa as coordenadas x e y.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

E se pudéssemos criar, separadamente pequenos trechos de código que fazem as seguintes etapas?

Atribui valores para as coordenadas x e y.

$$(a, b) + (c, d) = (a + c, b + d)$$

$$||(a, b)|| = \sqrt{a^2 + b^2}$$

Informa as coordenadas x e y.

De forma que esses trechos são descritos apenas uma vez, e podem ser chamados durante o código?

Informa as coordenadas x e y

O trecho que realiza essa etapa seria escrito como:

```
Receba um vetor, denominado exibir_v.

Escreva exibir_v.x

Escreva exibir_v.y

Encerre a rotina.
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Informa as coordenadas x e y

O trecho que realiza essa etapa seria escrito como:

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Informa as coordenadas x e y

O trecho que realiza essa etapa seria escrito como:

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Informa as coordenadas x e y

O trecho que realiza essa etapa seria escrito como:

```
procedure exibir (exibir_v: vetor)
        ► Exibe o valor de um vetor na tela
        write "(", exibir_v.x, ",", exibir_v.y, ")"
end procedure
```

Informa as coordenadas x e y

O trecho que realiza essa etapa seria escrito como:

Nome do procedimento, que é invocado no corpo do programa

```
procedure exibir (exibir_v: vetor)
        ► Exibe o valor de um vetor na tela
        write "(", exibir_v.x, ",", exibir_v.y, ")"
end procedure
```

Informa as coordenadas x e y

O trecho que realiza essa etapa seria escrito como:

Nome do procedimento, que é invocado no corpo do programa

Parâmetro que o programa recebe de entrada

```
procedure exibir (exibir_v: vetor)
        ► Exibe o valor de um vetor na tela
        write "(", exibir_v.x, ",", exibir_v.y, ")"
end procedure
```

Informa as coordenadas x e y

O trecho que realiza essa etapa seria escrito como:

Informa as coordenadas x e y

O trecho que realiza essa etapa seria escrito como:

Nome do procedimento, que é invocado no corpo do programa

Parâmetro que o programa recebe de entrada

```
procedure exibir (exibir_v: vetor)
    ▶ Exibe o valor de um vetor na tela
    write "(", exibir_v.x, ",", exibir_v.y, ")"
end procedure
```

Operações a serem executadas com os parâmetros de entrada

```
▶ Escrita dos dados de saída
    exibir(A)
    exibir(B)
    exibir(C)
    exibir(D)
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Informa as coordenadas x e y

O trecho que realiza essa etapa seria escrito como:

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Informa as coordenadas x e y

O trecho que realiza essa etapa seria escrito como:

```
▶ Escrita dos dados de saída
    exibir(A)
    exibir(B)
    exibir(C)
    exibir(D)
```

Nome do procedimento, que é invocado no corpo do programa

Parâmetro que o procedimento recebe de entrada

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Informa as coordenadas x e y

O trecho que realiza essa etapa seria escrito como:

```
procedure exibir (exibir_v: vetor)
        ► Exibe o valor de um vetor na tela
        write "(", exibir_v.x, ",", exibir_v.y, ")"
end procedure
```

```
► Escrita dos dados de saída
        exibir(A)
        exibir(B)
        exibir(C)
        exibir(D)
```

Nesse código:

O vetor exibir_v não foi alterado dentro da sub-rotina. Por essa razão, a sub-rotina **apenas recebe uma cópia** dos dados presentes no vetor exibir_v.

A sub-rotina não retorna nenhum valor de saída e, portanto, é chamada de **procedimento**.

Atribui valores para as coordenadas x e y.

O trecho que realiza essa etapa seria escrito como:

```
Receba um vetor, denominado receber_v,
além dos valores inteiros val_x e val_y.

Faça receber_v.x ← val_x

Escreva receber_v.y ← val_y

Encerre a rotina.
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Atribui valores para as coordenadas x e y.

O trecho que realiza essa etapa seria escrito como:

```
Receba um vetor, denominado receber_v,
além dos valores inteiros val_x e val_y.

Faça receber_v.x ← val_x

Escreva receber_v.y ← val_y

Encerre a rotina.
```

Nesse código:

A sub-rotina não retorna nenhum valor de saída e, portanto, é chamada de **procedimento**.

Os números inteiros val_x e val_y não são alterados dentro da sub-rotina. Por essa razão, a sub-rotina **recebe uma cópia** desses valores.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Atribui valores para as coordenadas x e y.

O trecho que realiza essa etapa seria escrito como:

```
Receba um vetor, denominado receber_v,
além dos valores inteiros val_x e val_y.

Faça receber_v.x ← val_x

Escreva receber_v.y ← val_y

Encerre a rotina.
```

Nesse código:

A sub-rotina não retorna nenhum valor de saída e, portanto, é chamada de **procedimento**.

Os números inteiros val_x e val_y não são alterados dentro da sub-rotina. Por essa razão, a sub-rotina **recebe uma cópia** desses valores.

O vetor receber_v será alterado dentro da sub-rotina. Por essa razão, a sub-rotina não pode recebe-lo por cópia. O que fazer?

Atribui valores para as coordenadas x e y.

O trecho que realiza essa etapa seria escrito como:

Receba um vetor, denominado receber_v, além dos valores inteiros val_x e val_y.

Faça receber_v.x ← val_x

Escreva receber_v.y ← val_y

Encerre a rotina.

O vetor receber_v será alterado dentro da sub-rotina. Por essa razão, a sub-rotina não pode recebe-lo por cópia. O que fazer?

Podemos passar um ponteiro, que recebe o endereço do vetor receber_v. Esse tipo de passagem é chamado **passagem por referência**.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Atribui valores para as coordenadas x e y.

O trecho que realiza essa etapa seria escrito como:

```
procedure receber (val_x: integer, val_y: integer, var receber_v: vetor)
        ► Recebe as coordenadas x e y e as coloca em um vetor.
        receber_v.x ← val_x
        receber_v.y ←val_y
end procedure
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Atribui valores para as coordenadas x e y.

O trecho que realiza essa etapa seria escrito como:

```
procedure receber (val_x: integer, val_y: integer, var receber_v: vetor)
    ► Recebe as coordenadas x e y e as coloca em um vetor.
    receber_v.x ← val_x
    receber_v.y ←val_y
end procedure
```

Operações a serem executadas com os parâmetros de entrada

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Atribui valores para as coordenadas x e y.

O trecho que realiza essa etapa seria escrito como:

Variáveis passadas por cópia

```
procedure receber (val_x: integer, val_y: integer, var receber_v: vetor)
    ► Recebe as coordenadas x e y e as coloca em um vetor.
    receber_v.x ← val_x
    receber_v.y ←val_y
end procedure
```

Operações a serem executadas com os parâmetros de entrada

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Atribui valores para as coordenadas x e y.

O trecho que realiza essa etapa seria escrito como:

Indicador de que a passagem ocorre por referência

Variáveis passadas por cópia

```
procedure receber (val_x: integer, val_y: integer, var receber_v: vetor)
    ► Recebe as coordenadas x e y e as coloca em um vetor.
    receber_v.x ← val_x
    receber_v.y ←val_y
end procedure
```

Operações a serem executadas com os parâmetros de entrada

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Atribui valores para as coordenadas x e y.

O trecho que realiza essa etapa seria escrito como:

Variáveis passadas por cópia

Indicador de que a passagem ocorre por referência

Variável passada por referência

```
procedure receber (val_x: integer, val_y: integer, var receber_v: vetor)
    ► Recebe as coordenadas x e y e as coloca em um vetor.
    receber_v.x ← val_x
    receber_v.y ←val_y
end procedure
```

Operações a serem executadas com os parâmetros de entrada

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Atribui valores para as coordenadas x e y.

O trecho que realiza essa etapa seria escrito como:

$$\|(a, b)\| = \sqrt{a^2 + b^2}$$

Construindo a sub-rotina acima, criando um procedimento com passagem de parâmetro por referência:

```
procedure norma_ref (calc_v: vetor, var norma: integer)
        ► Calcula a norma de um vetor, e a coloca em uma variável
        norma ← raiz((calc_v.x)^2 + (calc_v.y)^2)
end procedure
```

```
► Cálculo da norma dos vetores
      norma_ref(A, normaA)
      norma_ref(B, normaB)
```

$\|(a, b)\| = \sqrt{a^2 + b^2}$

O trecho que realiza essa etapa seria escrito como:

Receba um vetor, denominado calc_v.

Declare uma variável real, chamada norma.

Faça norma = $\sqrt{calc_v.x^2 + calc_v.y^2}$

Retorne, para o usuário, a variável norma.

Encerre a rotina.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

$\|(a, b)\| = \sqrt{a^2 + b^2}$

O trecho que realiza essa etapa seria escrito como:

Receba um vetor, denominado calc_v.

Declare uma variável real, chamada norma.

Faça norma = $\sqrt{calc_v.x^2 + calc_v.y^2}$

Retorne, para o usuário, a variável norma.

Encerre a rotina.

Já sabemos como escrever esses comandos

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

$\|(a, b)\| = \sqrt{a^2 + b^2}$

O trecho que realiza essa etapa seria escrito como:

Receba um vetor, denominado calc_v.

Declare uma variável real, chamada norma.

Faça norma = $\sqrt{calc_v.x^2 + calc_v.y^2}$

Retorne, para o usuário, a variável norma.

Encerre a rotina.

Já sabemos como escrever esses comandos

Como informar que o código deve retornar a variável norma para o usuário?

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

$$\|(a, b)\| = \sqrt{a^2 + b^2}$$

O trecho que realiza essa etapa seria escrito como:

```
function norma_func(calc_v: vetor): real
        ► Calcula a norma de um vetor, e a retorna
        declare:
                norma: real
        norma ← raiz((calc_v.x)^2 + (calc_v.y)^2)
        return norma
end function
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

$\|(a, b)\| = \sqrt{a^2 + b^2}$

O trecho que realiza essa etapa seria escrito como:

Nome da função, que é invocado no corpo do programa

```
function norma_func(calc_v: vetor): real
        ► Calcula a norma de um vetor, e a retorna
        declare:
                norma: real
        norma ← raiz((calc_v.x)^2 + (calc_v.y)^2)
        return norma
end function
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

$\|(a, b)\| = \sqrt{a^2 + b^2}$

O trecho que realiza essa etapa seria escrito como:

Nome da função, que é invocado no corpo do programa

Tipo de dado que é retornado pela função

```
function norma_func(calc_v: vetor): real
        ► Calcula a norma de um vetor, e a retorna
        declare:
                norma: real
        norma ← raiz((calc_v.x)^2 + (calc_v.y)^2)
        return norma
end function
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

$\|(a, b)\| = \sqrt{a^2 + b^2}$

O trecho que realiza essa etapa seria escrito como:

$\|(a, b)\| = \sqrt{a^2 + b^2}$

O trecho que realiza essa etapa seria escrito como:

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

$\|(a, b)\| = \sqrt{a^2 + b^2}$

O trecho que realiza essa etapa seria escrito como:

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

$\|(a, b)\| = \sqrt{a^2 + b^2}$

O trecho que realiza essa etapa seria escrito como:

```
function norma_func(calc_v: vetor): real
        ▶ Calcula a norma de um vetor, e a retorna
        declare:
                norma: real
        norma ← raiz((calc_v.x)^2 + (calc_v.y)^2)
        return norma
end function
```

```
▶ Cálculo da norma dos vetores
        normaA ← norma_func(A)
        normaD ← norma_func(D)
```

Nesse código:

A sub-rotina retorna um valor de saída e, portanto, é chamada de **função**.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

$$(a, b) + (c, d) = (a + c, b + d)$$

O trecho que realiza essa etapa seria escrito como:

```
function soma(V1: vetor, V2:vetor): vetor
        ► Calcula a soma de dois vetores, e a retorna
        declare:
                r_soma: vetor
        r_soma.x ← V1.x + V2.x
        r_soma.y ← V1.y + V2.y


        return r_soma
end function
```

```
► Cálculo da soma dos vetores
        C ← soma(A,B)
        D ← soma(A,C)
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

Na construção da sub-rotina, variáveis que recebem valores por referência devem ser declaradas com a palavra var antes do nome da variável. Variáveis que recebem valores por cópia devem ser declaradas apenas pelo seu identificador e tipo.

Dentro da sub-rotina, todas as variáveis são operadas apenas pelo seu identificador, campos e índices, da mesma forma que no código principal.

Ao chamar uma sub-rotina, as variáveis de qualquer tipo devem ser passadas por seu identificador, seja por cópia ou por referência.

Procedimentos e funções podem receber qualquer número de valores, seja por referência ou por cópia.

Procedimentos não possuem nenhum valor de retorno.

Funções possuem apenas um valor de retorno.

Uma função pode retornar qualquer tipo de dado.

Variáveis criadas dentro de uma sub-rotina só podem ser utilizadas dentro dessa sub-rotina.

Compete ao programador, ao implementar o algoritmo em uma dada linguagem, utilizar os ponteiros e endereços de forma adequada para realizar a passagem por referência.

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

"Construa um programa que receba as três notas de um aluno em uma disciplina. Em seguida, o programa deve imprimir um boletim, contendo cada uma das notas e a média final."

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

```
Algorithm: boletim
Input: notas de um aluno em 3 provas.
Output: boletim contendo nota de cada prova e sua média final
► Declaração das variáveis utilizadas
declare:
        notas[3]: real
        media: real

► Leitura das notas
for i → 0 to 2 step 1 do
        read notas[i]
        media ← notas[i]
end for
► Determinação da média
media ← media / 3
► Escrita dos resultados
for i → 0 to 2 step 1 do
        write notas[i]
end for
write media
end Algorithm
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

```
define:
        total_notas: integer  6
```

FUNDAMENTOS DO PENSAMENTO ALGORÍTMICO

```
Algorithm: boletim
Input: notas de um aluno em 3 provas.
Output: boletim contendo nota de cada prova e sua média final
► Declaração das variáveis utilizadas
define:
        total_notas: integer  6
declare:
        notas[total_notas]: real
        media: real

► Leitura das notas
for i → 0 to (total_notas – 1) step 1 do
        read notas[i]
        media ← notas[i]
end for
► Determinação da média
media ← media / total_notas
► Escrita dos resultados
for i → 0 to (total_notas – 1) step 1 do
        write notas[i]
end for
write media
end Algorithm
```